Desain Protokol pada Lapisan Aplikasi untuk Pemantauan Kualitas Udara melalui Jaringan TCP/IP

Kartika Firdausy[1], Yudi Utomo Imardjoko[2], dan Bambang Nurcahyo Prastowo[3]

[1] Program Studi Teknik Elektro, Fakultas Teknologi Industri, Universitas Ahmad Dahlan, Jl. Prof. Dr. Soepomo, Janturan, Yogyakarta

[2] Fakultas Teknik, Universitas Gadjah Mada, Yogyakarta

[3] Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada, Yogyakarta

Abstract

Air quality monitoring is one part of air pollution control. In this research, Internet technology is utilized for that monitoring. Communications through Internet need protocols, which comprises various connection layers, from hardware layer up to the application layer. In the mechanism of the observed air quality monitoring, an application protocol that capable in handling the transfer of measurement data from monitoring stations to the central station is required. Central station also delivers the processed data to the public, as well as sends commands to the monitoring station whenever a change in measurement setting has to be made. For the sake of connection efficiency, only the server has to be connected to the Internet all over the time. Clients are only connected to the Internet while sending measurement data. The server is the central station and there are two kind of clients, monitoring station client and user client.

The objective of this research is to design an application layer protocol for air quality monitoring, that is a communication protocol between the server and monitoring station clients, called Simple Remote Measurement Protocol (SRMP). The communication between the server and user clients uses HTTP (Hypertext Transfer Protocol). Apache HTTP Server is used for the web server. The gathered data are processed by the server and stored in a database. The HTTP clients may query a data from the server. The implementation of SRMP protocol uses Java programming language, while the database uses MySQL. PHP is used to create dynamic web page and JPGraph is used to produce data plots.

Program prototype testing has already performed using data that are simulated based on the actual measured value. Testing was also done by running the program on the Internet. Interoperability was examined by running the program on different operating system: Windows and Linux.

Keywords *: application protocol, air quality monitoring, client-server architecture.*

1. Pendahuluan

Berbagai dampak negatif dapat ditimbulkan oleh semburan asap kendaraan bermotor dan pembuangan berbagai macam limbah industri. Misalnya, karbon monoksida (CO) yang masuk ke dalam tubuh manusia dapat menyebabkan gangguan pada fungsi darah (Wardhana, 1999). Paru-paru yang terkontaminasi NO_2 dapat mengalami peradangan sehingga penderita sulit bernafas yang dapat mengakibatkan kematian. Polusi udara juga mengancam kesehatan anak-anak dan bayi (Reinberg, 2001).

Masalah pencemaran udara ini perlu ditangani secara berkesinambungan. Salah satu bagian pengendalian pencemaran udara adalah pemantauan terhadap kualitas udara. Dengan pemantauan secara terus-menerus dan secara langsung, tindakan preventif dan antisipatif seandainya terjadi emisi bahan pencemar udara yang melebihi batas ambang dapat dilakukan

ISSN : 0216 - 7565

dengan segera. Untuk itu perlu dirancang suatu mekanisme pemantauan yang dapat beroperasi secara otomatis serta terus-menerus yang menggunakan teknologi komputer serta memanfaatkan teknologi Internet.

Agar dapat saling berkomunikasi, diperlukan adanya suatu protokol komunikasi dalam suatu jaringan komputer, dalam hal ini adalah komputer di stasiun pusat dengan komputer di stasiun pemantau dan dengan komputer publik. Pada mekanisme pemantauan kualitas udara yang diteliti ini, diperlukan protokol yang mampu menangani pengiriman data hasil pemantauan dari stasiun pemantau ke stasiun pusat untuk dikelola. Stasiun pusat selain menerima data dari stasiun pemantau juga mengirimkan data yang sudah dikelola kepada masyarakat umum, serta mengirimkan perintah ke stasiun pemantau jika diperlukan adanya perubahan *setting* pemantauan, misalnya laju pengukuran atau laju pengiriman data.

Pada penelitian ini, stasiun pusat bertindak sebagai *server* yang selalu menunggu permintaan dari klien, dalam hal ini adalah pengiriman data dari stasiun pemantau dan permintaan data dari masyarakat umum. *Server* juga memiliki kemampuan untuk mengirimkan perintah kepada klien, yaitu perintah kepada stasiun pemantau, serta mengirimkan data kepada masyarakat umum (pengguna). Di sini, hanya *server* yang harus selalu terhubung ke Internet, klien terhubung ke Internet pada saat-saat tertentu saja, sehingga biaya koneksi dapat ditekan. Untuk selanjutnya, yang disebut *server* adalah stasiun pusat. Pada skema ini, terdapat dua jenis klien, yaitu klien stasiun pemantau, yang melakukan pengukuran konsentrasi parameter dasar kualitas udara, dan klien pengguna, yaitu masyarakat umum yang mengakses informasi pemantauan kualitas udara.

Protokol aplikasi yang sudah tersedia dipandang tidak dapat mengakomodasi keperluan tersebut, sehingga perlu dirancang sebuah protokol pada lapisan aplikasi yang sesuai untuk kebutuhan pemantauan kualitas udara, yaitu protokol komunikasi antara *server* dan klien stasiun pemantau, yang selanjutnya akan disebut *Simple Remote Measurement Protocol* (SRMP). HTTP (*Hypertext Transfer Protocol*) digunakan untuk komunikasi antara *server* dan klien pengguna.

Ada tiga cara yang dapat dipilih jika membutuhkan sebuah protokol pada lapisan aplikasi untuk digunakan pada sebuah permasalahan tertentu (Rose, 2001), yaitu menggunakan protokol aplikasi yang sudah ada, mengembangkan model protokol di atas protokol aplikasi yang sudah ada, membuat protokol yang sama sekali baru. Pilihan pertama adalah yang paling mudah. Tetapi sesungguhnya tidak mudah untuk mendapatkan sebuah protokol aplikasi yang benar-benar sesuai dengan semantik protokol yang dibutuhkan.

Protokol yang dirancang oleh de Albuquerque dan Lelièvre-Berna (1998), yang diberi nama *Instrument Data Transfer Protocol* (IDTP), merupakan sebuah protokol lapisan aplikasi yang digunakan untuk pemantauan difraktometer neutron melalui Internet. Protokol IDTP dibuat di atas lapisan transpor UDP (*User Datagram Protocol*) pada *port* 7001, menggunakan arsitektur klien-*server*. Pada protokol ini tidak dijumpai adanya mekanisme pengiriman perintah dari *server* ke klien.

Penelitian yang dilakukan oleh Magrabi, dkk (1998) bertujuan untuk membuat sebuah sistem pemantauan elektrokardiogram berbasis *web*. Sistem ini melakukan pemantauan elektrokardiogram pasien yang dirawat di rumah sehingga dokter dapat selalu memantau kondisinya dari mana saja. Data hasil pemantauan di sisi klien berukuran cukup besar dan dikirim ke *server* melalui protokol FTP (*File Transfer Protocol*). *Server* dapat diakses oleh *browser web* melalui protokol HTTP (*Hypertext Transfer Protocol*), tetapi pengiriman data elektrokardiogram dari *server* memakai FTP. Pada sistem ini tidak dijumpai adanya mekanisme pengiriman perintah dari *server* ke klien.

Pengelolaan kualitas air sungai secara terintegrasi menggunakan teknologi Internet telah dilakukan oleh Cianchi, dkk (2000). Sistem ini sangat kompleks dan lebih menekankan pada penggunaan agen cerdas untuk pemantauan kualitas air sungai.

Arpaia, dkk (1997) melakukan penelitian tentang pengukuran jarak jauh untuk eksperimen di laboratorium sekolah (*Remote Laboratory - RemLab*). Dengan memakai RemLab, para siswa (praktikan) dapat melakukan eksperimen pengoperasian instrumen, misalnya osiloskop digital, dari jarak jauh dengan cara mengakses *server* memakai *browser web*. Sebagai *server* adalah sebuah komputer yang dihubungkan dengan instrumen tersebut melalui antarmuka IEEE-488.

Metode yang diusulkan oleh Orr dan Miller (1998) untuk pemantauan lingkungan adalah dengan menggunakan satelit, pesawat pemantau, helikopter dan sensor darat untuk keperluan akuisisi data, kemudian mengirimkan data tersebut ke stasiun pusat. Internet dipakai untuk komunikasi antar stasiun pusat pemantauan.

Baxter, Jr. (2000) mengajukan usulan sebuah sistem peringatan dini deteksi kondisi lingkungan. Data yang dikumpulkan oleh satelit dikirim ke *server* basis data. Klien pengguna dapat mengakses data tersebut melalui Internet.

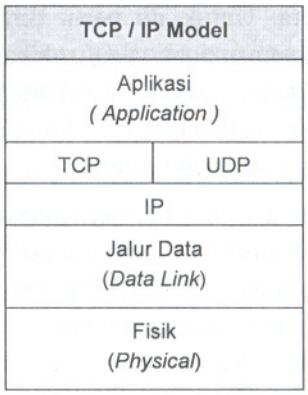
2. Fundamental

Pemantauan kualitas udara

Untuk memberikan kemudahan dan keseragaman informasi kualitas udara kepada masyarakat di lokasi dan waktu tertentu serta sebagai bahan pertimbangan dalam melakukan upaya-upaya pengendalian pencemaran udara, disusun Indeks Standar Pencemar Udara (ISPU). ISPU ditetapkan dengan cara mengubah kadar pencemar udara yang terukur menjadi suatu angka yang tidak berdimensi (MENLH, 1997). Parameter-parameter dasar ISPU meliputi partikulat (PM_{10}), sulfurdioksida (SO_2), karbonmonoksida (CO), ozon (O_3), nitrogendioksida (NO_2)

Konsep dasar jaringan

Jaringan adalah sekumpulan komputer dan perangkat lain yang dapat saling mengirim dan menerima data (Harold, 1997). Arsitektur jaringan dengan model referensi TCP/IP terlihat pada Gambar 1.



Gambar 1. Model Lapisan Referensi TCP/IP (Feit, 1997)

Protokol komunikasi

Untuk keperluan pengiriman dan penerimaan data dalam jaringan diperlukan suatu protokol, yaitu kesepakatan dari semua pihak yang berkomunikasi tentang bagaimana komunikasi tersebut harus dilaksanakan (Tanenbaum, 2000).

Pelanggaran pada protokol akan membuat komunikasi menjadi terhambat atau bahkan bisa menggagalkan komunikasi. Terdapat berbagai macam protokol yang mendefinisikan berbagai aspek dalam komunikasi jaringan. Misalnya HTTP (*Hypertext Transfer Protocol*), mendefinisikan komunikasi antara *browser web* dan *server*. Standar protokol dipublikasikan secara luas, sehingga perangkat keras dan perangkat lunak yang berasal dari produsen yang berbeda bisa saling berkomunikasi.

Desain protokol aplikasi

Dalam mendesain sebuah protokol aplikasi, perlu diperhatikan beberapa mekanisme yang umumnya dimiliki oleh sebuah protokol aplikasi, yaitu antara lain (Rose, 2001) :

a. mekanisme pembingkaian (*framing*), yang menjelaskan cara pembatasan awal dan akhir setiap pesan yang ditransfer;
b. mekanisme pengkodean (*encoding*), yang berhubungan dengan bagaimana sebuah pesan direpresentasikan saat dikirimkan;

c. mekanisme pelaporan kesalahan (*error reporting*), yaitu mekanisme untuk menyampaikan informasi tentang kesalahan yang mungkin terjadi oleh salah satu pihak kepada pihak lain sehingga dapat dilakukan penanganan yang tepat terhadap kesalahan tersebut;

d. mekanisme *multiplexing*, berhubungan dengan tiga mekanisme, yaitu paralelisme (kemampuan protokol untuk melakukan pengolahan beberapa permintaan secara bersamaan dengan cara *multithreading*), kendali aliran, dan segmentasi (memecah sebuah pesan menjadi beberapa bagian yang lebih kecil dan dikirimkan secara bersamaan);

e. mekanisme autentikasi (*user authentication*), yaitu usaha untuk mengidentifikasi dengan siapa suatu pihak melakukan koneksi. Autentikasi juga dilakukan untuk meyakinkan bahwa pesan yang dikirimkan adalah pesan yang sebenarnya;

f. mekanisme keamanan pengiriman (*transport security*), dilakukan dengan cara mengenkripsi data yang dikirimkan.

3. Metodologi

Dalam penelitian ini dilakukan perancangan protokol aplikasi antara *server* stasiun pusat dan klien stasiun pemantau. Rancangan protokol diimplementasikan dengan pembuatan prototip program, yaitu berupa prototip yang meliputi perangkat lunak *server* stasiun pusat dan perangkat lunak di sisi klien stasiun pemantau.

Perangkat lunak yang digunakan adalah Java Development Kit 1.3 dari Sun Microsystem, MySQL 3.23 untuk pembuatan basis data, *browser* yang mendukung Java (Microsoft Internet Explorer 4.0 atau lebih, Netscape Communicator 4.x), perangkat lunak untuk pembuatan halaman *web* (Netscape Composer, Microsoft FrontPage), PHP 4.1.2 digunakan untuk pembuatan halaman *web* dinamik, dan JPGraph digunakan untuk pembangkitan tampilan grafik secara dinamik.

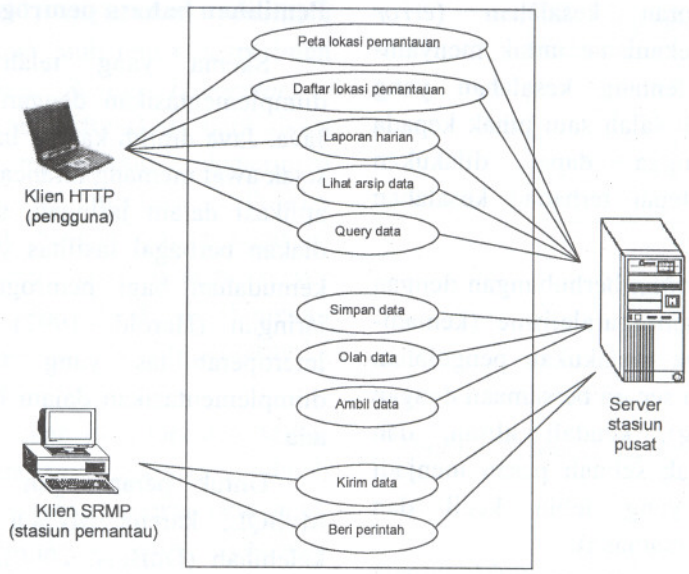
Pemilihan bahasa pemrograman

Skema yang telah disusun kemudian diimplementasikan dengan bahasa pemrograman Java. Java dipilih karena bahasa pemrograman ini sejak awal memang dirancang untuk pemrograman aplikasi dalam jaringan, sehingga telah menyediakan berbagai fasilitas yang akan memberikan kemudahan bagi pemrograman aplikasi dalam jaringan (Harold, 1997). Java juga memiliki interoperabilitas yang tinggi, karena dapat diimplementasikan dalam berbagai platform yang ada.

Untuk perancangan basis data digunakan MySQL, karena MySQL memiliki kelebihan-kelebihan (DuBois, 2000), di antaranya: konektivitas tinggi (sesuai untuk aplikasi jaringan), dapat diperoleh secara gratis (*open distribution*), memiliki kecepatan tinggi dan relatif mudah digunakan.

Untuk pembuatan halaman *web* dinamik digunakan PHP, yaitu pengolah skrip yang dapat menghasilkan halaman informasi dengan format HTML. PHP juga mempunyai kemampuan untuk mengolah permintaan dari klien serta berhubungan dengan *server* basis data untuk memperoleh data yang bersifat dinamik. Basis data yang didukung oleh PHP antara lain adalah Oracle, dBase, DB2, dan MySQL (Bakken, 2001). Untuk pembangkitan tampilan grafik secara dinamik digunakan JPGraph.

Pengujian prototip dilakukan dengan menghubungkan *server* stasiun pusat dan klien stasiun-stasiun pemantau melalui Internet. Pengujian dilakukan untuk mengevaluasi unjuk kerja prototip yang dibuat. Semula pengujian direncanakan dengan menggunakan data-data sekunder hasil pemantauan kualitas udara yang diperoleh dari instansi terkait, yaitu Badan Pengendalian Dampak Lingkungan Daerah (Bapedalda) Propinsi DIY dan Balai Teknik Kesehatan Lingkungan (BTKL) Yogyakarta. Tetapi karena pemantauan yang dilakukan oleh BTKL dan Bapedalda dilakukan setahun sekali, sehingga belum sesuai dengan aturan yang tercantum dalam keputusan Kepala Bapedal (BAPEDAL, 1997), maka dalam penelitian ini dipakai simulasi data yang disesuaikan dengan contoh data yang diperoleh



Gambar 2. Diagram Fungsionalitas Sistem

untuk menguji protokol aplikasi yang dirancang. Selain itu, contoh data juga diperoleh dari situs-situs Internet tentang pemantauan kualitas udara.

Perancangan protokol aplikasi diawali dengan mendefinisikan fungsi dari masing-masing bagian yang ada dalam sebuah mekanisme pemantauan kualitas udara. Hal ini diilustrasikan dalam Gambar 2. Pada gambar tersebut diberikan fungsi masing-masing aktor atau pihak yang ada dalam sistem yang dirancang. Setelah mengetahui apa saja yang akan dilakukan oleh sistem yang dirancang, perlu ditentukan protokol aplikasi apa yang dapat melaksanakan fungsi-fungsi tersebut.

Fungsi-fungsi yang dilaksanakan antara klien pengguna dan *server* stasiun pusat dilakukan dengan mekanisme permintaan-respon (*request-response*). Hal ini dapat ditangani oleh protokol HTTP. Fungsi-fungsi pengolahan, penyimpanan, dan pengambilan data dilakukan secara internal di dalam *server* stasiun pusat. Fungsi-fungsi yang dilakukan antara klien stasiun pemantau dan *server* stasiun pusat terjadi secara dua arah, karena data dikirimkan oleh stasiun pemantau kepada stasiun pusat sedangkan perintah diberikan oleh stasiun pusat kepada stasiun pemantau. Protokol yang akan dirancang ini, disebut *Simple Remote Measurement Protocol* (SRMP), terletak pada lapisan aplikasi sehingga pada lapisan di bawahnya tetap menggunakan protokol TCP/IP.

Mekanisme pembingkaian yang dipilih adalah *octet-stuffing* sehingga pemrosesan dapat dilakukan dengan lebih mudah. Kelemahan *octet-stuffing* yang lebih lambat tidak terlalu menjadi masalah karena data yang dikirimkan berukuran kecil. Karakter batas yang dipilih adalah CR-LF. Pengkodean baris pesan dilakukan dengan menggunakan format yang tersusun atas sebuah kata kunci (*keyword*) dan parameter yang masing-masing dibatasi oleh karakter spasi, sebagaimana format berikut ini:

```
Kata_kunci Parameter1 Parameter2 ...
```

Kata kunci menunjukkan operasi atau permintaan yang diberikan oleh klien kepada *server* sedangkan parameter merupakan nilai yang menyertai kata kunci tersebut. Parameter dapat berjumlah satu atau lebih, namun dimungkinkan pula adanya kata kunci tanpa parameter. Kata kunci yang disediakan dalam rancangan protokol *Simple Remote Measurement Protocol* (SRMP) adalah: **ID** (identitas), **PASS** (*password*), **DATA**, **CMND** (*command*), **NOOP** (*no operation*), dan **QUIT**

Contoh penulisan pesan adalah sebagai berikut:

```
ID Tugu
```

di mana **ID** adalah kata kunci untuk memberikan identitas sedangkan **Tugu** adalah nama stasiun pemantau yang terletak di daerah Perempatan Tugu.

Data parameter dasar kualitas udara yang diberikan yang merangkai kata kunci **DATA** berisi informasi tanggal dan waktu pengukuran, kode parameter dasar kualitas udara serta nilai hasil pengukurannya. Semua informasi tersebut disusun dalam satu baris pesan dengan dibatasi oleh spasi. Kode parameter dasar kualitas udara adalah : PM10, SO2, CO, O3, dan NO2. Konsentrasi parameter kualitas udara diukur dalam satuan µg/m3.

Contoh penulisan pesan untuk mengirimkan data adalah:

```
DATA 2002-3-19 10:15:00 NO2 25.7
```

di mana tanggal dan waktu pengukuran adalah 19 Maret 2002 pukul 10:15, parameter dasar kualitas udara yang diukur adalah nitrogendioksida (NO_2) yang bernilai 25.7 µg/m^3.

Mekanisme pelaporan kesalahan dilakukan dengan memberi respon positif yang diawali dengan +**OK** atau respon negatif diawali dengan -**ERR**, diikuti oleh keterangan yang mendukung respon tersebut. Contohnya, apabila klien memberikan *password* yang salah, maka *server* akan menjawab dengan pesan:

```
-ERR Password salah, ulangi kata kunci
ID atau QUIT
```

Pada penelitian ini, data yang dikirimkan oleh klien stasiun berukuran kecil sehingga tidak perlu disediakan mekanisme *multiplexing*. Autentikasi pengguna dilakukan dengan cara mengirimkan *username* berupa identitas stasiun pemantau dan *password*.

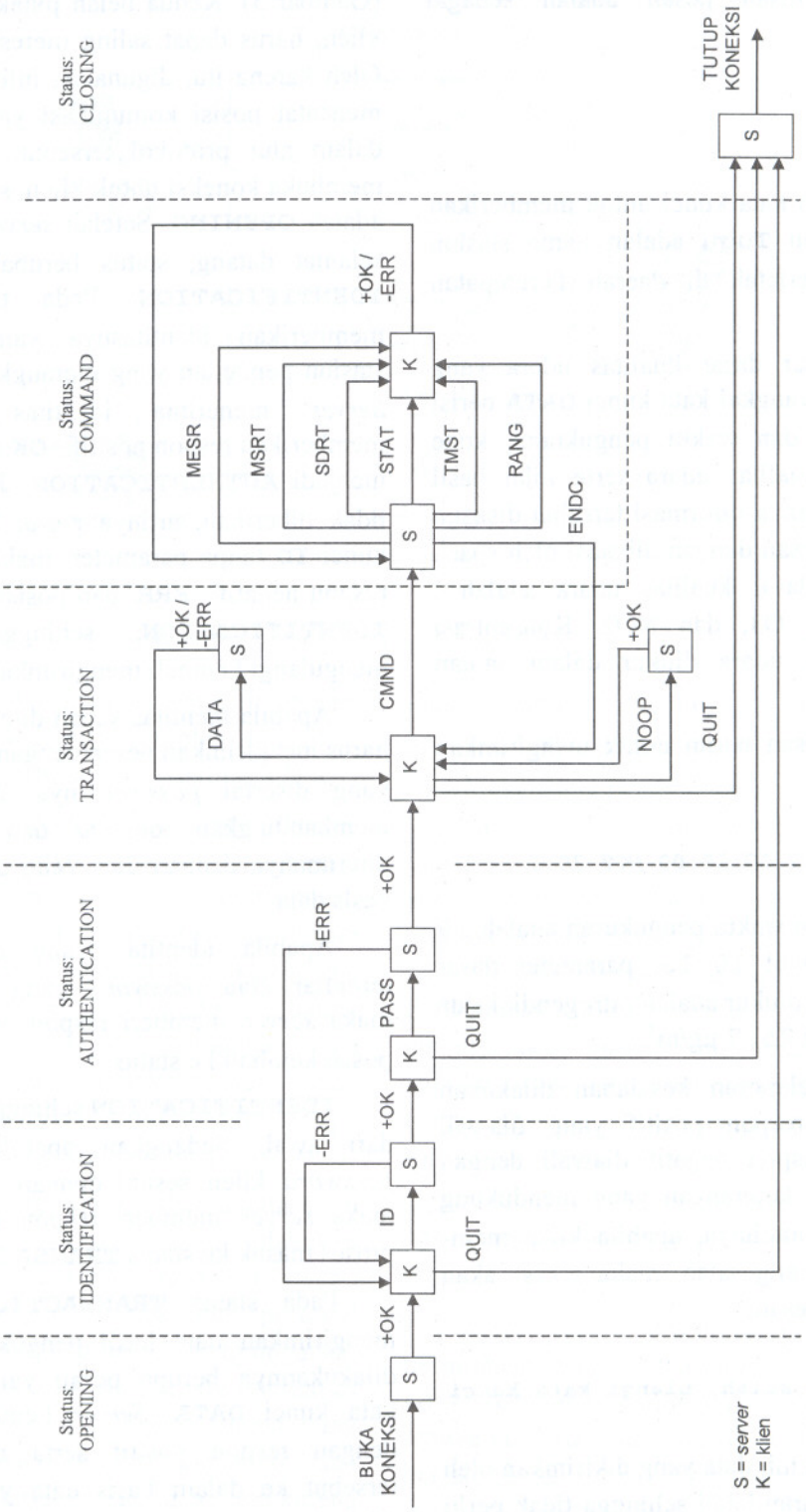
Aliran komunikasi antara *server* dan klien diilustrasikan dalam diagram protokol SRMP (Gambar 3). Kedua belah pihak, yaitu *server* dan klien, harus dapat saling merespon dengan tepat. Oleh karena itu, digunakan informasi status yang mencatat posisi komunikasi yang sedang terjadi dalam alur protokol tersebut. Pada saat *server* membuka koneksi untuk klien, status yang dipakai adalah **OPENING**. Setelah *server* memberi pesan selamat datang, status berubah menjadi status **IDENTIFICATION**. Pada tahap ini, klien memberikan identitasnya yang berupa nama stasiun pemantau yang merangkai kata kunci **ID**. *Server* menerima identitas tersebut serta memberikan respon positif +**OK** dan status berubah menjadi **AUTHENTICATION**. Jika nama stasiun tidak diberikan, artinya pesan hanya berisi kata kunci **ID** tanpa parameter, maka *server* memberi respon negatif -**ERR**. dan posisi tetap pada status **IDENTIFICATION**, sehingga klien harus mengulangi kembali mengirimkan identitasnya.

Apabila identitas sudah diterima *server*, klien harus mengirimkan pesan dengan kata kunci **PASS** yang disertai *password*-nya. *Server* kemudian membandingkan identitas dan *password* yang diterimanya dengan data yang dimilikinya dalam basis data.

Apabila identitas yang dikirimkan tidak terdaftar atau *password* yang diberikan salah, maka *server* memberi respon negatif -**ERR** dan posisi kembali ke status

IDENTIFICATION sehingga proses diulangi dari awal. Sedangkan apabila identitas dan *password* klien sesuai dengan daftar yang ada, maka *server* memberi respon positif +**OK** dan proses masuk ke status **TRANSACTION**.

Pada status **TRANSACTION**, klien dapat mengirimkan data hasil pengukuran yang telah dilakukannya berupa pesan yang menggunakan kata kunci **DATA**. *Server* kemudian menjawab dengan respon positif serta menyimpan data tersebut ke dalam basis data yang dimilikinya. Oleh karena setiap stasiun pemantau biasanya melakukan pengukuran terhadap banyak parameter dasar kualitas udara, maka pengiriman data dapat dilakukan tidak hanya satu kali.



Gambar 3. Diagram Protokol Aplikasi SRMP yang Dirancang

Setelah semua data terkirim, maka klien memberi kesempatan kepada *server* untuk memberi perintah kepada dirinya. Hal ini dilakukan dengan mengirimkan pesan dengan kata kunci **CMND** dan status berubah menjadi **COMMAND**. Pada tahap ini *server* mendapat giliran untuk mengirim permintaan dan klien merespon. *Server* akan menentukan apakah ada perintah yang harus diberikan kepada klien stasiun pemantau untuk melakukan sesuatu. Apabila ada, maka *server* mengirimkan perintah tersebut. Karena klien tidak selalu terhubung ke *server*, jika *server* perlu mengirim perintah kepada klien maka perintah-perintah tersebut disimpan terlebih dahulu. Pada saat klien menghubungi *server*, perintah-perintah tersebut akan dikirimkan secara berurutan kepada klien.

Dalam rancangan ini, protokol memiliki beberapa perintah yang umumnya diberikan kepada stasiun pemantau, yaitu : **MESR** (*measure*), **MSRT** (*measurement rate*), **SDRT** (*send rate*), **STAT** (*status*), **TMST** (*time to start*), **RANG** (*measurement range*), **ENDC** (*end command*). Format perintah ini sama dengan format kata kunci yang diberikan oleh klien, yaitu rangkaian perintah dan parameter yang dibatasi oleh spasi. Klien menerima perintah tersebut dan memberi respon positif +**OK** apabila berhasil melaksanakannya dan memberi respon negatif – **ERR** apabila terjadi kegagalan.

Apabila perintah yang diberikan oleh *server* adalah **ENDC**, maka aliran protokol kembali ke posisi di mana klien berperan sebagai pihak yang mengirimkan permintaan (*request*) dan *server* merespon, yaitu status **TRANSACTION**. Apabila klien bermaksud untuk menutup koneksi, maka dikirimkan pesan dengan kata kunci **QUIT**, masuk ke status **CLOSING** dan *server* akan merespon secara positif serta kemudian memutuskan koneksi TCP.

Setelah merancang protokol aplikasi yang diperlukan, langkah berikutnya adalah membuat prototip program yang mengimplementasikan protokol tersebut. Prototip ini terdiri atas program pada *server* stasiun pusat, program klien stasiun pemantau, dan rancangan halaman *web* yang akan menampilkan data hasil pengukuran dan peta lokasi pemantauan di *browser* klien pengguna.

4. Hasil dan Pembahasan

Program *Server* pada stasiun pusat menangani protokol HTTP dan SRMP, serta menangani transaksi data ke sebuah *server* basis data MySQL. Mengingat komunikasi dengan klien pengguna melalui protokol HTTP sudah ditangani oleh program Apache, maka program di *server* yang dibuat adalah program untuk mengimplementasikan protokol SRMP.

Pada prototip program *server* yang dibuat, terdapat sebuah kelas utama yang diberi nama **ServerSRMP** serta sebuah kelas pendukung untuk menangani koneksi dengan klien stasiun pemantau, yaitu kelas **KoneksiSRMP**.

Pemilihan nomor *port* untuk protokol SRMP dapat dilakukan dengan pertimbangan bahwa nomor *port* tersebut belum digunakan untuk layanan standar. Yang penting nomor *port* ini harus disepakati dengan klien stasiun pemantau sehingga komunikasi dapat berjalan dengan baik. Apabila ada klien yang menghubungi *server*, maka program membuat *socket* baru untuk berkomunikasi dengan klien tadi, kemudian membuat *instance* kelas **KoneksiSRMP** dengan *socket* tersebut sebagai parameternya dan menjalankannya. Selanjutnya, *server* kembali menunggu permintaan koneksi berikutnya.

Agar dapat menangani banyak permintaan dari beberapa klien secara simultan, maka kelas **KoneksiSRMP** merupakan turunan dari kelas **Thread**. Kelas ini menggunakan *socket* yang dibuat oleh *server* tadi untuk merespon klien. Sesuai dengan protokol yang telah dirancang, kelas ini memanfaatkan informasi status untuk mengetahui posisi komunikasi yang sedang terjadi dalam alur protokol yang sedang berjalan. Oleh karena itu, pada kelas ini didefinisikan status yang akan dipakai sepanjang terjadinya komunikasi antara *server* dengan klien SRMP.

Cara menyimpan data ke dalam basis data dilakukan dengan membuka koneksi ke *server* MySQL dan mengirimkan perintah SQL di dalam *method* **SimpanDataPengukuran**.

Pada program klien di stasiun pemantau hanya terdapat sebuah kelas, yaitu **Pemantau** yang menangani akuisisi data dari sensor serta pengiriman data ke *server* menggunakan protokol SRMP. Secara umum, program utama dari kelas ini adalah membaca *file* konfigurasi dan kemudian memeriksa waktu pada saat itu. Jika waktu pengukuran sudah terlampaui maka dilakukan akuisisi data dari sensor menggunakan *method* **Akuisisi** dan kemudian data yang diperoleh dikirimkan ke *server* SRMP menggunakan *method* **KirimData**.

Sesuai dengan persyaratan yang tercantum dalam Surat Keputusan Kepala Bapedal Nomor Kep-107/Kabapedal/1997 tentang Pedoman Teknis Perhitungan dan Pelaporan serta Informasi Indeks Standar Pencemar Udara, maka informasi paling utama yang disediakan untuk dipublikasikan kepada masyarakat harus memuat waktu pelaporan, ketentuan waktu, lokasi, ISPU setiap parameter yang diukur, ISPU maksimum, parameter pencemar kritis, kategori ISPU, serta gambar kategori dan rentang ISPU. (BAPEDAL, 1997). Oleh karena itu halaman laporan harian merupakan bagian yang terpenting dari prototip halaman *web* yang dibuat dalam penelitian ini. Selain itu, dalam prototip yang dibuat juga disediakan halaman *web* lain, seperti: peta dan daftar stasiun pemantau, hasil pengukuran terbaru pada setiap stasiun pemantau baik dalam bentuk teks maupun grafis.

Halaman-halaman tersebut berupa halaman statik dan halaman yang mengandung informasi yang dinamik, misalnya data pengukuran terbaru dan informasi yang melibatkan perhitungan. Halaman dinamik ini dibuat dalam *file* berekstensi php.

Uji coba di jaringan

Prototip program *server* dan klien telah diuji di jaringan lokal maupun jaringan Internet, yaitu pada *server* **s2.te.ugm.ac.id** dan **t-fisika.ugm.ac.id**. Kedua *server* tersebut terhubung ke jaringan Internet melalui sambungan *leased line*. Prototip klien stasiun pemantau juga telah dicoba dijalankan melalui sambungan *dial-up* menggunakan modem dan saluran telepon. Hasil pengujian menunjukkan bahwa secara umum, kedua prototip program tersebut telah berfungsi dengan baik.

Gambar 4 menunjukkan contoh sesi komunikasi antara *server* stasiun pusat dan klien stasiun pemantau yang terjadi saat pengiriman data hasil pengukuran ketika dilakukan pengujian.

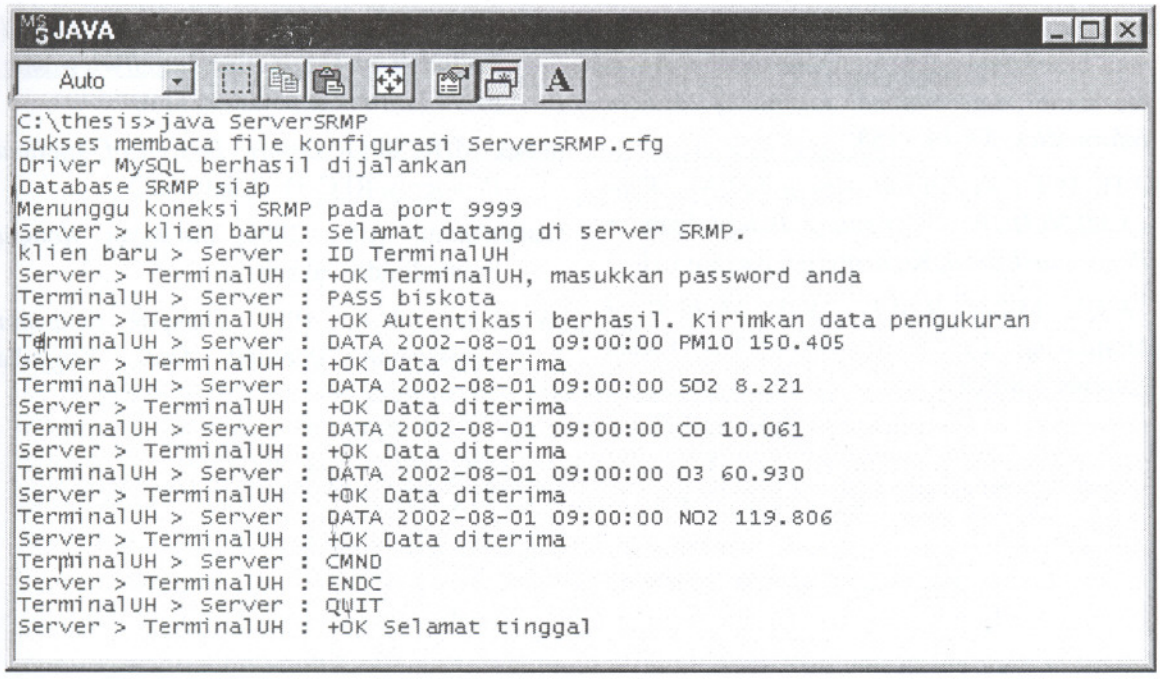
Pada pengiriman data dari stasiun pemantau menggunakan sambungan *dial-up*, sistem operasi diatur agar secara otomatis mengaktifkan modem untuk menghubungi ISP ketika ada permintaan untuk membuka koneksi ke *server* melalui sebuah *socket*. Setelah terhubung dengan *server*, data dikirim. Sistem operasi kemudian secara otomatis memutuskan koneksi ke ISP setelah *socket* tersebut ditutup dan tidak ada aktivitas transfer data selama beberapa saat.

5. Kesimpulan

Dari penelitian yang telah dilakukan dapat ditarik kesimpulan bahwa telah berhasil dibuat :

1. Rancangan protokol pada lapisan aplikasi untuk komunikasi antara *server* stasiun pusat dan klien stasiun pemantau kualitas udara, yang disebut *Simple Remote Measurement Protocol* (SRMP);
2. Prototip program *server* stasiun pusat yang mengimplementasikan protokol SRMP;
3. Prototip halaman *web* yang ditampilkan pada *browser* di klien pengguna;
4. Pengujian prototip-prototip di atas menggunakan jaringan lokal dan jaringan Internet.

ISSN : 0216 - 7565



Gambar 4. Sesi Komunikasi Antara *Server* Stasiun Pusat dengan Klien Stasiun Pemantau

Ucapan Terimakasih

Penulis mengucapkan terima kasih kepada Balai Teknik Kesehatan Lingkungan (BTKL) Yogyakarta dan Badan Pengendalian Dampak Lingkungan Daerah (Bapedalda) Propinsi DIY yang telah mengijinkan penulis untuk memperoleh dan menggunakan data serta referensi yang penulis perlukan.

Daftar Pustaka

Arpaia, P., Baccigalupi, A., Cennamo, F., Daponte, P., 1997, "A Remote Measurement Laboratory for Educational Experiments", *Measurement, 21,* No.4, 157-169.

Bakken, S.S., Aulbach, A., Schmid, E., Winstead, J., Wilson, L.T., Lerdorf, R., Suraski, Z., Zmievski, A., and Ahto, J., 2001, *PHP Manual*, PHP Documentation Group.

BAPEDAL, 1997, *Keputusan Kepala Bapedal Nomor Kep-107/Kabapedal/ 1997 tentang Pedoman Teknis Perhitungan dan Pelaporan serta Informasi ISPU*, Bapedal, Jakarta.

BAPEDALDA, 2001, *Laporan Pemantauan Kualitas Udara di DIY Tahun 2001*, Bapedalda Propinsi DIY, Yogyakarta.

Baxter, Jr., J.F., 2000, *Early Warning Detection and Notification Network for Environmental Conditions*, US Patent Number 6,023,223.

BTKL, 1997, *Hasil Pemeriksaan Kualitas Udara Daerah Padat Lalu-lintas Kota Yogyakarta*, BTKL, Yogyakarta.

Cianchi, P., Marsili-Libelli, S., Burchi, A., Burchielli, S., 2000, "Integrated River Quality Management using Internet Technologies", *Watermatex 2000*, 18–20 Sept.2000, Gent, Belgia.

Cornell, G. dan Hortsmann, C. S., 1997, *Core Java*, Penerbit Andi, Yogyakarta.

De Albuquerque, M.P. dan Lelievre-Berna, E., 1998, "Remote monitoring over the Internet", *Nuclear Instruments and Methods in Physics Research*, A, 412, 140-145.

DuBois, P., 2000, *MySQL*, New Riders Publishing, Indianapolis.

Feit, S., 1997, *TCP/IP : Architecture, Protocols, and Implementations with IPv6 and IP Security*, 2nd ed., McGraw-Hill, New York.

Harold, E.R., 1997, *Java Network Programming*, O'Reilly, Sebastopol.

Magrabi, F., Lovell, N.H., Celler, B.G., 1999, "A web-based approach for ECG monitoring in the home", *International Journal of Medical Informatics, 54,* 145-153.

MENLH, 1997, *Keputusan Meneg LH No. Kep-45/MENLH/10/1997 tentang Indeks Standar Pencemar Udara*, Kementerian Negara LH.

Orr, W.W., Miller, R.M.P., 1998, *Method for Monitoring the Environment*, US Patent Number 5,808,916.

Reinberg, S., 2001, *Infant Deaths Linked to Pollution in U.S. Cities*, http://www.who.int/peh/ceh/articles/ pollution.htm.

Rose, M.T., 2001, *On the Design of Application Protocols*, RFC 3117, InterNIC

Tanenbaum, A.S., 2000, *Jaringan Komputer*, Prenhallindo, Jakarta.

Wardhana, W.A., 1999, *Dampak Pencemaran Lingkungan*, Penerbit Andi, Yogyakarta.

ISSN : 0216 - 7565